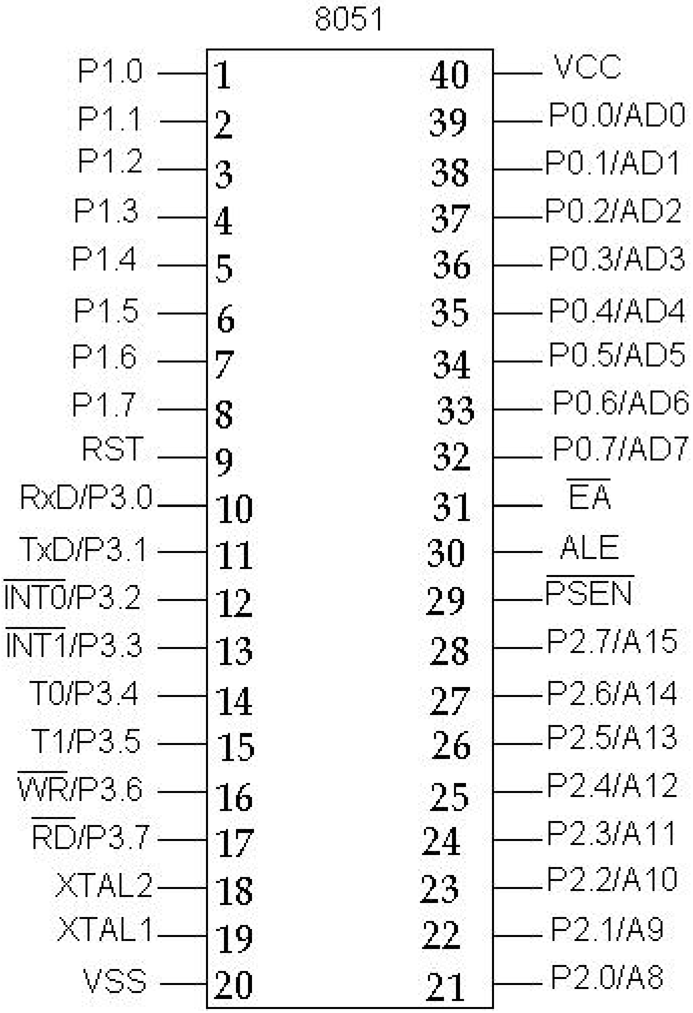
Bashir Oyetunji
University of Saskatchewan
IEEE SPARC 2006-07

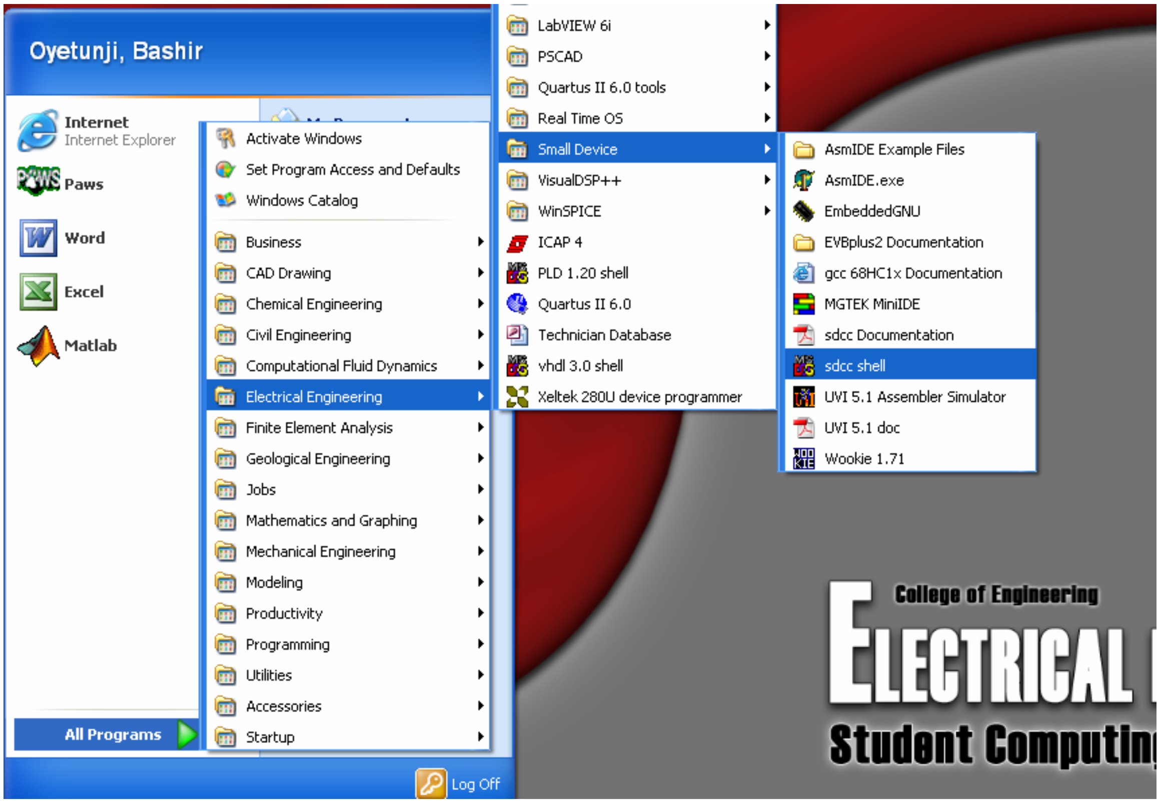
Complete 8051 Guide

Pinouts



Procedure to compile and burn code onto the 8051

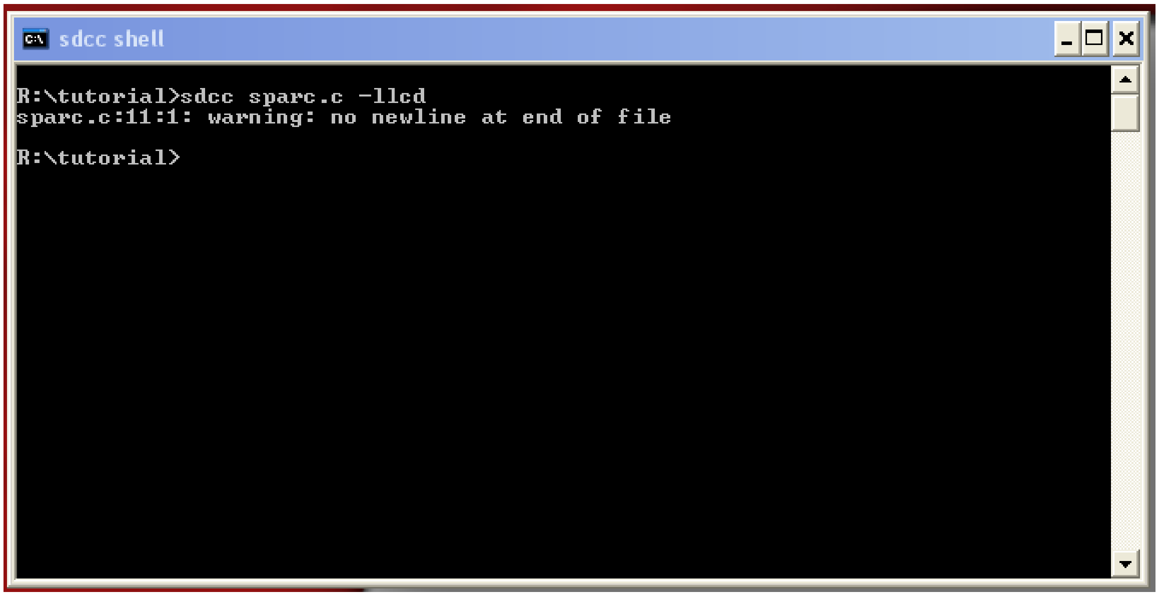
1) Open up the sdcc shell



2) Change directory into the working directory that contains your code

3) Type in the command line:

```
    sdcc filename.c –llcd
```



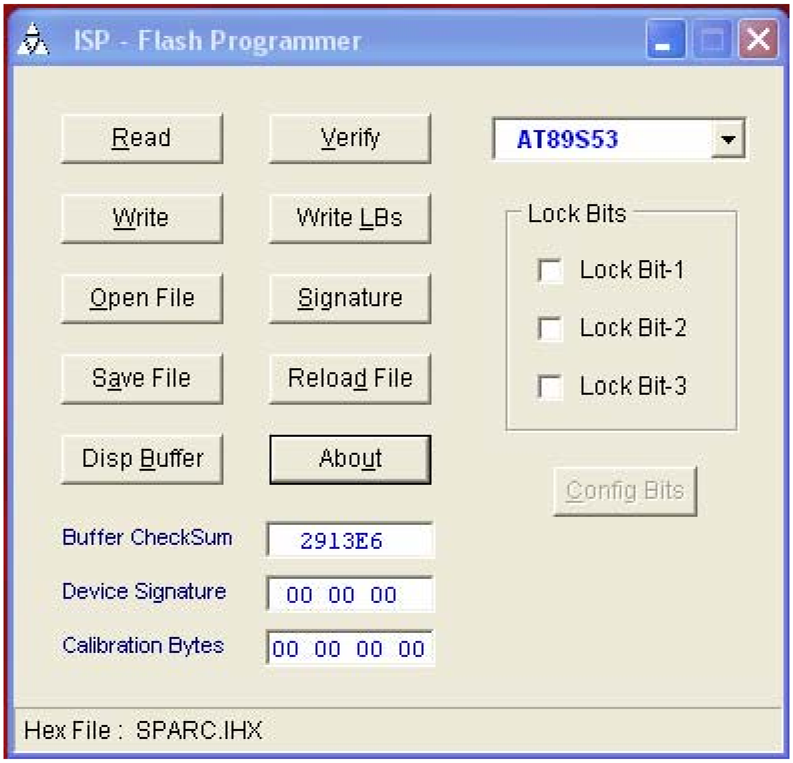
```
R:\tutorial>sdcc sparc.c -llcd
sparc.c:11:1: warning: no newline at end of file

R:\tutorial>
```

The –llcd option at the end is to include the lcd library. “-l” is for include and “lcd” is for the library name.

The compiler sometimes gives the “no newline at end of file” warning as shown below. You can just ignore it. A successful compilation produces a couple of files, we are interested in the .ihx file. That is what you burn onto the microcontroller.

4) Open the burning program by typing burn8953 in the sdcc shell. That opens up the Atmel ISP – Flash Programmer.



AT89S53

Buffer CheckSum 2913E6

Device Signature 00 00 00

Calibration Bytes 00 00 00 00

Hex File : SPARC.IHX

5) Select your device type, which is the AT89S53 for this case.

6) Click on Open File and locate the hex file you want to burn. Remember to select “All Files (*.*)” in the Files of type field.
Hex files: .hex, inh(intel hex) and ihx.
Select the filename.ihx file.

7) Write the hex file to the micro by clicking on Write. A successful write will display the “Programming/Verify OK” message.

SAMPLE CODE

1) LCD PROGRAM

This program does nothing other than to display a message on the LCD display. “8051.h” is the header file to include all microcontroller functions. “stdio.h” is the standard input/output library. Without it, the printf wont work. The LCD library uses Port 0 on the micro. If you need Port 0 for anything else, don’t compile with –llcd and don’t have any printfs in your code.

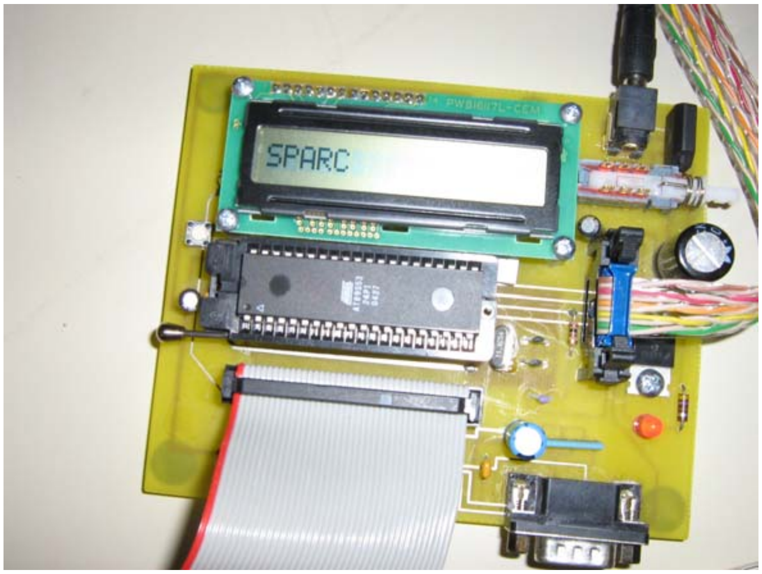
```
/******************************************
*
*      Program displays “SPARC” on the LCD
*
*
******************************************/

#include <8051.h>
#include <stdio.h>


void main(void) {
      printf("SPARC\n");

      // loop forever
      for (;;) {
        // do nothing for now
      }

}
```



2) Input/Output Ports

This program shows how to work with reading and writing to ports and individual pins. To set Pin3 of Port2 high, use the command:

```
      P2_3=1;
```

To set Pin6 of Port1 low, use:

```
      P1_6=0;
```

To set all the pins on port2 high, use:

```
P2=0xFF; // don’t use P2=1, that will only set P2_0 high
            // and P2=0x00; will set pins on port2 low
```

The following command wil set pins 7,6,4,3,2 high and 5,1,0 low on port2.

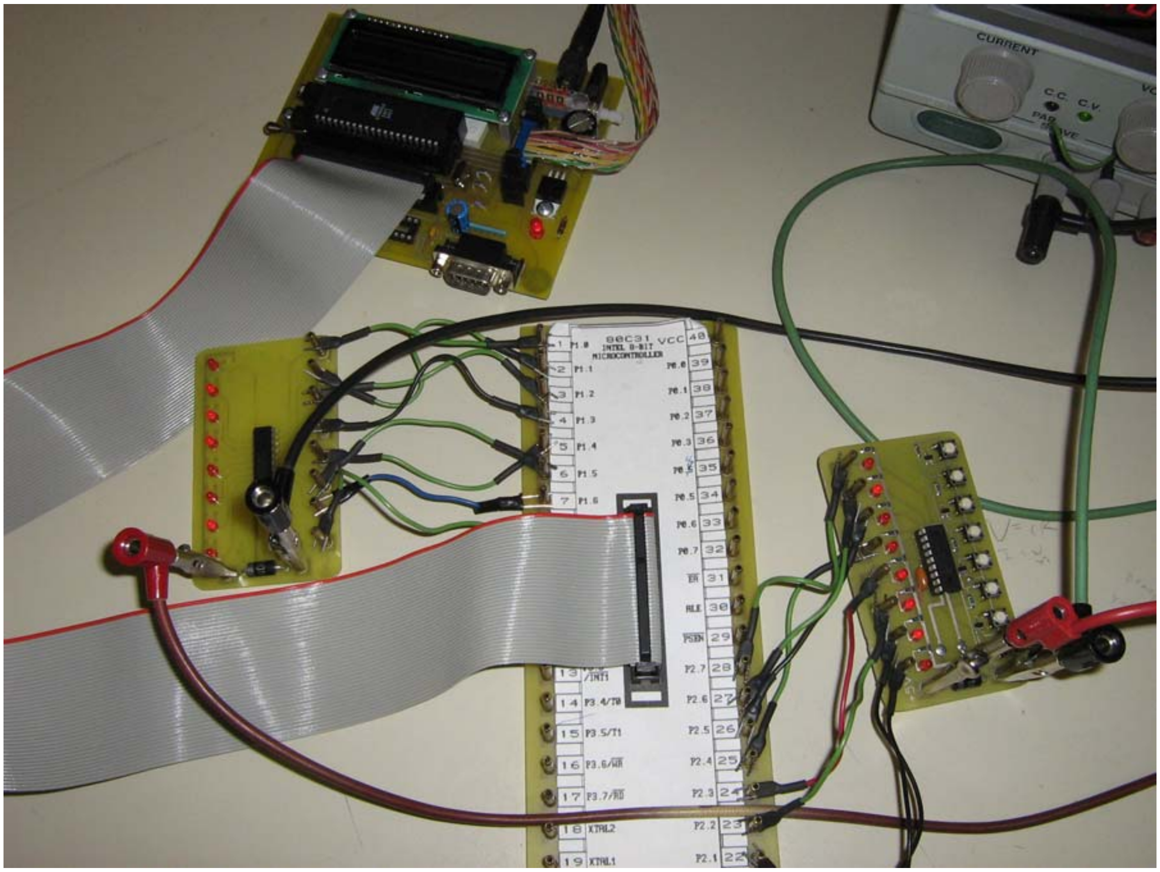
```
P2=0xDC; //DC in hex = 11011100 in binary
```

```
/*******************************************
*
*     Program maps the status of Port2's pins
*     onto Port1
*
********************************************/


#include <8051.h>
#include <stdio.h>


void main(void) {
      printf("SPARC\n");

      // loop forever
      for (;;) {
            P1=P2;  //P1 stands for Port1
        /*****alternate code******/
      /*    P1_0=P2_0;  //P1_0 stands for Pin 0 of Port1
            P1_1=P2_1;
            P1_2=P2_2;
            P1_3=P2_3;
            P1_4=P2_4;
            P1_5=P2_5;
            P1_6=P2_6;
            P1_7=P2_7;
                              */
      }
}
```



3) External Interrupts

This program sets and uses the external interrupts. External interrupt 0 corresponds to interrupt 0(pin 12), and external interrupt 1(pin 13) corresponds to interrupt 2.

```
/*******************************************
*
*     Program initializes both external interrupts
*     (interrupts 0 and 2)
*     When External int0 is pushed on, port 1 is set low
*     When External int1 is pushed on, port 1 is set high
*
*******************************************/


#include <8051.h>
#include <stdio.h>


void main(void) {
      // Interrupt Initialization
      IT0 = 1;                // make ext. interrupt 0 edge triggered
      IT1 = 1;                // make ext. interrupt 1 edge trigerred
      IE0 = 0;
      EX0 = 1;                // enable external interrupt 0
      EX1 = 1;                // enable external interrupt 1
      EA  = 1;                // global interrupts enable

      printf("SPARC\n");
      // loop forever
      for (;;) {
            // do nothing
      }
}


void EXO_int0 (void) interrupt 0 {
      printf(“Int 1 occurred\n”);
}

void EX1_int2 (void) interrupt 2 {
      printf(“Int 2 occurred\n”);
}
```

4) Timer Program

The program is a simple clock that uses the overflow timer interrupt. Note that the code is written for the 12MHz clocked micro. Read the code comments and modify it if you change crystals. The program has an accurate delay function. Call delay(y) to create a y*100usecs delay, where y is an integer; i.e. delay(10000) will create a 1 second delay.

```
/********************************************
*
*     Program sets up the 16-bit overflow timer interrupt
*     It is a simple seconds counter
*     Timer is set to interrupt every 100usecs timer
*     The program has an accurate delay function
*
********************************************/


#include <8051.h>
#include <stdio.h>

int i;
int secs;
void delay(int);

void main(void) {
      //initialize all variables
      i=0;
      secs=0;
      // Interrupt Initialization
      IT0 = 1;                // make ext. interrupt 0 edge triggered
      IT1 = 1;                // make ext. interrupt 1 edge trigerred
      IE0 = 0;
      EX0 = 1;                // enable external interrupt 0
      EX1 = 1;                // enable external interrupt 1
      EA  = 1;                // global interrupts enable

/*
      crystal - 12Mhz
      internal clock is divide by 12 = 12Mhz/12= 1Mhz
      clk cycle = 1/1Mhz=1us
      --------------------
      chose to use 100usecs timer
      Time of timer = 2^16 - 100us = 65536 - 100 = 65436 dec = FF9C hex
      high order = FF, low order = FC
*/

//    Timer Initialization
      TMOD = 0x01;      // sets timer to a 16 bit continuous timer
      TH0  = 0XFF;      // Initialize timer0 MSB (timer for 100us)
      TL0  = 0X9C;            // Initialize timer0 LSB
      ET0  = 1;               // Enable Timer0 overflow interrupt
      TR0  = 1;               // Start Timer0

      printf("SPARC\n");
      P1=0;
      P2=0;
      // loop forever
      for (;;) {
            /*when the counter reaches 10000, that is equivalent to 1
            second
            10000 counts of 100usecs= 1second
            increment secs variable and print new time*/
            if (i>=10000) {
                  secs++;
                  printf("%d secs\n", secs);  //display time in seconds
                  i=0;
            }
      }
}



// Timer ISR (interrupt service routine) is activated every 100us
void Timer0_ISR (void) interrupt 1 {
      // reset timer
      TH0  = 0XFF;            //reset timer MSB
      TL0  = 0X9C;            //reset timer LSB
      i++;  // global variable i is incremented every 100us
}
```

```
// delay function, delay=100usecs*cycles
void delay(int cycles) {
      i=0;
      while (i<cycles) {
            i=i;
      }
    i=0;

}
```

5) LCD Scroller

This program was written by me and Erik during SPARC 2004-05. We were learning how to program the 8051, and we figured we’d test if pointers worked in sdcc. I guess they do.

Set the variable “displayLen” to the number of characters you can print on your LCD display. Send the function ScrollString a string or character array longer than the length of displayLen, and the text will scroll through your display.

```
/*SPARC 2004
Bashir Oyetunji, Erik Kulyk
*/

/* include following sdcc header file */
#include <8051.h>
#include <stdio.h>

/* defines */
#define IN_PORT P0      /* define a whole port */
#define OUT_PORT P2      /* define a whole port */
#define displayLen 16   /* length of the display */

// function declaration
void program();
void delay();
void scrollString(char*);

void main(void) {
      IN_PORT = 0xFF; //set IN_PORT High to Enable Data Reading
    OUT_PORT = 0x00; //set OUT_PORT Low

 // Loop the following code forever
      for(;;) {
                  program();
            }

}


void program(void) {
      scrollString("This is the longest word in the world + 1");
}
```

```
void scrollString(char* word) {
      char *pos;  // character pointers
      char *adder;
      int i;
      pos=word;





      // while you havent reached the end of the string
      while (*pos != '\0') {
            i=0;
            adder=pos;
            /* print character by character from the current
            location till the end of the string or till you
            print the number of characters your LCD
            supports (displayLen)
            */
            while ((i<displayLen)&&(*adder!='\0')) {
                  printf("%c", *adder);
                  adder++;

                  i++;
            }
            // delay so you can read it
            delay();
            //clear display at the end of string
            printf("\n");
            pos++;

      }
}

/*
 * Simple delay loop
 * Easy to write, but using timer would be better
 */
void delay (void)
{
    unsigned char i, j, k;

    i = 2;
    do {
        j = 0;
        do {
            k = 0;
            do {
            } while (--k);
        } while (--j);
    } while (--i);
}
```

6) Serial Transmitter

This and the next example are for serial communication on the microcontroller. This example writes a number to the transmit pin of the serial port, starting at 0 and incrementing, every time an external interrupt occurs. Watch the LCD for status information.

Example 7 is the code on the other micro that listens for data on the serial port and displays received characters onto the LCD display.

These examples were very helpful to the final year students last year that used the 8051 microcontroller for serial communication. They just took this code and went from there.

Note:
a) The grounds on both micros should be connected together (or to the same ground) to ensure they are working off the same reference value. The transmit pin(pin 11) on one micro is connected to the receive pin(pin 10) on the other, and vice versa.

b) Both micros should use the same crystal oscillator. Using 12MHz crystals actually produced a 10.15kbaud serial signal (was measured) instead of the required 9600 baud. As long as you transmit and receive at the same baud rate, you are fine. There will be a problem though talking to other serial devices that expect 9600 baud. To generate 9600 baud and other standard baud rates, swap the crystal with a 11.059MHz crystal. Then you are portable and your micro can talk to other serial devices and computers.

```
// Bashir Oyetunji
// September 2005

#include <8051.h>
#include <stdio.h>

int send;
void serialbuff(short int);

/*Function includes a 40ms delay*/
void delay (void)
{
    unsigned char i, j, k;

    i = 4;
    do {
        j = 0;
        do {
            k = 0;
            do {
            } while (--k);
        } while (--j);
    } while (--i);
}
```

```
void main() {
      send=0;

            // Interrupt Initialization
      IT0 = 1;                // make interrupt edge triggered
      IE0 = 0;


      SCON = 0x50;      /* uart in mode 1 (8 bit), REN=1 */
      PCON = 0x00;      // SMOD=0
      TMOD = 0x21;      //Timer 1 in Mode 2
      TH1 = 0xFD;       /* 9600 Bds at 11.059MHz */
      IE = 0x91;        //enable interrupts, and external interrupt 0
      TR1 = 1;          // Timer 1 run

      printf("Ready Tx\n");
      while (1);
}

void EXO_int (void) interrupt 0 {
      send++;
      printf("sending %X\n",send);
      delay();
      delay();
      delay();
      delay();
      delay();
      serialbuff(send);

}
/* Function called when SBUF is used. It waits for TI to be clear.
Once clear, the value is loaded to SBUF.
No further transmission takes place until TI is set*/

void serialbuff(short int val){
      SBUF = val;
      if (!TI) printf("sent %X\n",val);
      TI=0;
}
```

7) Serial Receiver

This is the receiving code to be burned on the other micro. It prints onto the LCD the single character data it receives from the serial port. It can read from any device sending RS232 data at 9600 baud, including reading data from a computer. You can connect it to a computer through the serial port of the development board, through a serial cable, through a null modem. The null modem crosses the transmit and receive lines between the micro and the computer. Just load up the hyperterminal program on your computer.

```
// Bashir Oyetunji
// September 2005

#include <8051.h>
#include <stdio.h>

char uart_data;

void delay (void)
{
    unsigned char i, j, k;

    i = 4;
    do {
        j = 0;
        do {
            k = 0;
            do {
            } while (--k);
        } while (--j);
    } while (--i);
}

void main (void) {
            // Interrupt Initialization
      IT0 = 1;                // make interrupt edge triggered
      IE0 = 0;


      SCON = 0x50;      /* uart in mode 1 (8 bit), REN=1 */
      PCON = 0x00;      // SMOD=0
      TMOD = 0x21;      //Timer 1 in Mode 2
      TH1 = 0xFD;       /* 9600 Bds at 11.059MHz */
      IE = 0x91;        //enable interrupts, and external interrupt 0
      TR1 = 1;          // Timer 1 run

      printf("Ready Rx\n");
      while (1);

}



void serial_IT(void) interrupt 4
{

      if (RI == 1)
      {                                        /* if reception occur */
        RI = 0;                          /* clear reception flag for
next reception */
        uart_data = SBUF;          /* Read receive data */
        printf("R %X\n", uart_data);
      }
      else printf("error\n");                  /* if emission occur */
                                          /* clear emission flag for
next emission*/
```

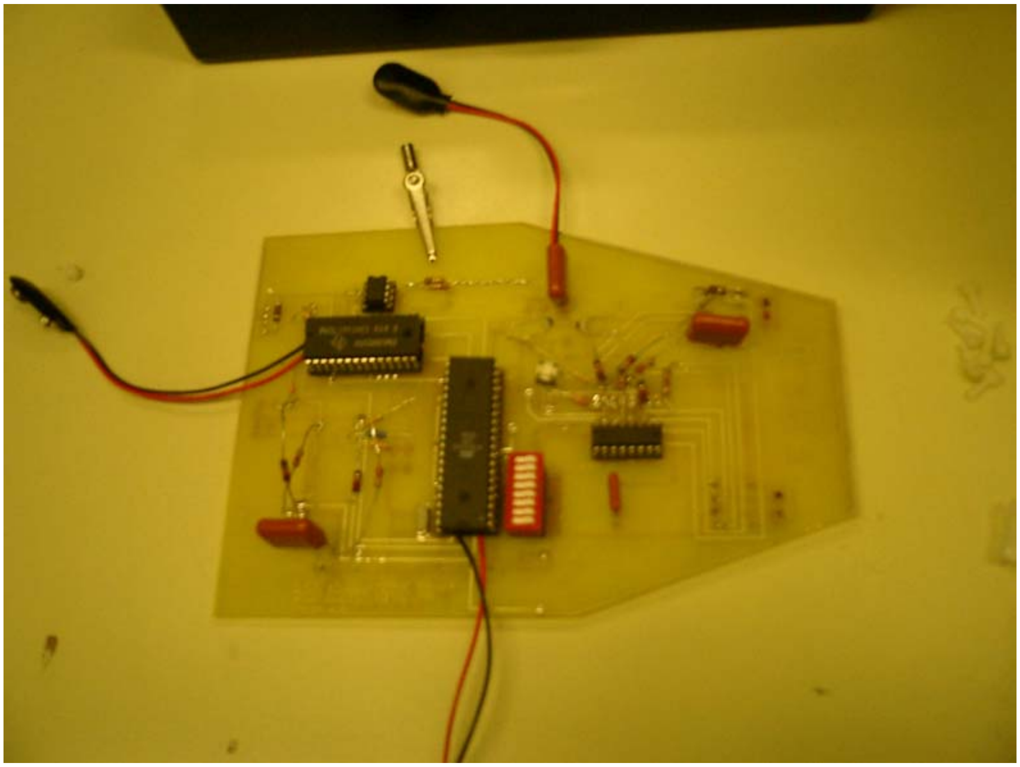
}

8) Functional Robotic Exploration Device

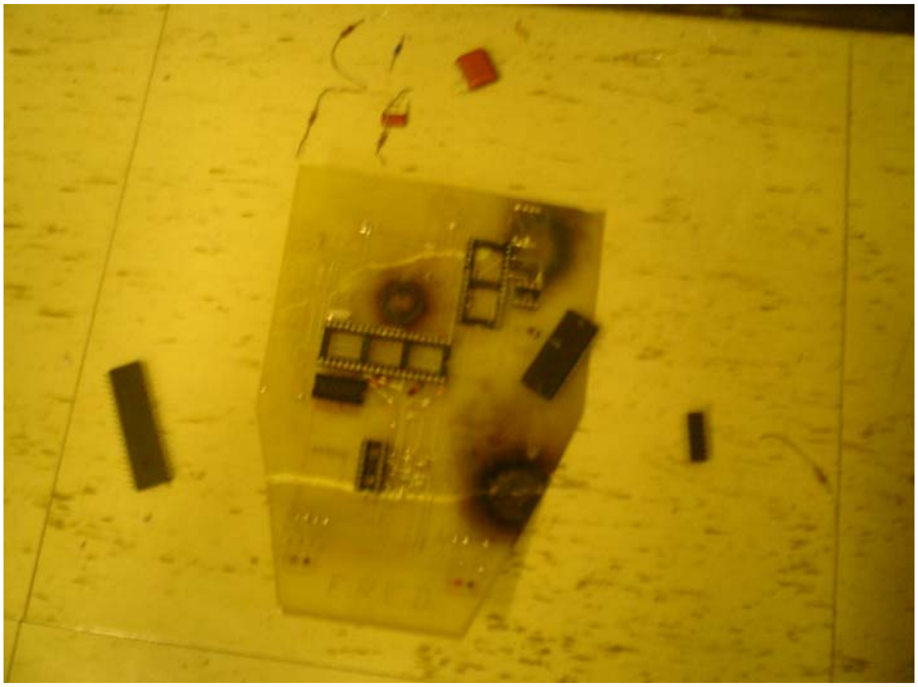
The FRED project was our EE323 (Electronic instrumentation) class project. The idea was to design a robot that you could set its distance from the wall (10cm<=distance<=50cm), and follow the wall at that distance with an accuracy of +/-1cm. It should be able to turn with the wall and avoid obstructions ahead.

The implementation required two side sensors that give analog data we had to convert to digital using the ADC (ADC0808 8-bit, 8-channel), and a sensor in the front that went low if there was an object with about 22cm and high otherwise.

PCB BEFORE

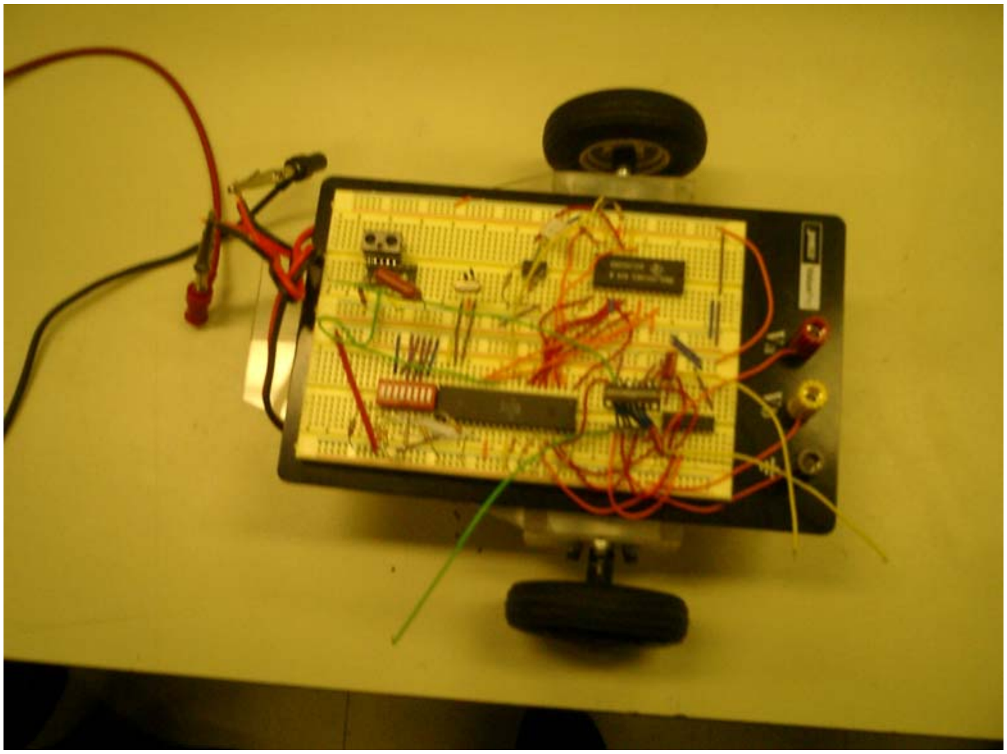


PCB AFTER



Whoever said smashing it would fix it, was so wrong.

FRED-FINISHED



The code worked perfectly and met all specifications. Googe video “FRED (Functional Robotic Exploration Device)” for the video of the robot. You can contact me or any of these guys if you have questions about the project.

```
/* EE323 Design Group
   Group Name - "Fred"
   Albert La
   Bashir Oyetunji
   Bryan Hamilton
   Corwin Head
   Herbert Mueller
   Stephen Tao
   Tim Fretz

   Last Modified Dec 5th 2005
   */

/* include following sdcc header file */
#include <8051.h>


/* defines */
int i;  // gets incremented every 100us, used to read from ADC
bit wall;   // wall being followed
unsigned int DIP_READ;        // value from DIP switch
unsigned int ADC_READS1;      // value from ADC input 0 (right sensor)
unsigned int ADC_READS2;      // value from ADC input 1 (left sensor)
unsigned int right_dist;      // look up value of ADC_READS1
unsigned int left_dist;       // look up value of ADC_READS2

// function declaration
void program();         // main program
void dist_read(); //read distance
void delay(int);  // delay=100usecs*int
void delay_secs(int);   // delay for n seconds
void delay_blinkR_secs(int);  // delay for n seconds and blink R_LED
void delay_blinkL_secs(int);  // delay for n seconds and blink L_LED
void R_forward(); // right motor forward
void R_backward();      // right motor backward
void R_off();           // right motor off
void L_forward(); // left motor forward
void L_backward();      // left motor backward
void L_off();           // left motor off
void search();          // search state of vehicle
void merge_right();     // turn right slightly
void merge_left();      // turn left slightly
void turn_right(int);   // turn right sharp
void turn_left(int);    // turn left sharp
void go_forward();            // go forward
void go_backward();           // go backward
unsigned int calc_distS1(unsigned int);  // look up table for right
sensor
unsigned int calc_distS2(unsigned int);   // look up table for left
sensor
```

```
// motor controls
#define L_ENABLE  P0_0
#define L_DIRECTION P0_1
#define R_ENABLE  P0_2
#define R_DIRECTION P0_3

// LED controls
#define R_LED P2_0
#define L_LED P2_7
#define LED_LOW 1 // LED is off when pin is high
#define LED_HIGH 0      // LED is on when pin is low

// WALLS
#define RIGHT_WALL 1
#define LEFT_WALL 0

// FRONT Sensor
#define FRONT_SENSOR P3_1



void main(void) {
      unsigned int DIP_INL;   // low order byte of DIP switch
      unsigned int DIP_INH;   // high order byte of DIP switch

      // read the high order BCD, save in DIP_INH
      DIP_INH = P2;
      DIP_INH = DIP_INH&0x70; // convert high order to BCD by ANDing
with 0111 0000
      DIP_INH=DIP_INH>>4;

      // read the low order BCD, save in DIP_INL
      DIP_INL = P2;
      DIP_INL = DIP_INL&0x0E; // convert low order to BCD by ANDing
with 0000 1110

      // compute input value at P2 (Binary->BCD->Decimal), save in
DIP_READ
      DIP_READ=(DIP_INH*10) + DIP_INL;

//Variable Initializations
      i=0;  // reset timer counter to 0

      // Interrupt Initialization
      IT0 = 1;                // make interrupt edge triggered
      IE0 = 0;
      EX0 = 1;                // enable external interrupt 0
      EA  = 1;                // global interrupts enable
```

```
/*
      crystal - 12Mhz
      internal clock is divide by 12 = 12Mhz/12= 1Mhz
      clk cycle = 1/1Mhz=1us
      --------------------
    choose to use 100usecs timer
      Time of timer = 2^16 - 100us = 65536 - 1 = 65436 dec = FF9C hex
      high order = FF, low order = FC

*/

// Timer Initialization
      TMOD = 0x01;            // sets timer to a 16 bit continuous
timer
      TH0  = 0XFF;            // Initialize timer0 MSB (timer for
100us)
      TL0  = 0X9C;            // Initialize timer0 LSB
      ET0  = 1;               // Enable Timer0 overflow interrupt
      TR0  = 1;               // Start Timer0

      // reset all ADC control pins
      P3_5=0;           // set ALE low
      P3_4=0;           // set address bit low
      delay(4);  // give adc enough time (400us) to turn off

      R_off();    // turn right and left motors off initially
      L_off();

      search();

      // Loop the following code forever
      for(;;)
            {
                  program();
            }
}


// continuous motor operation
void program(void) {
      dist_read();  // read sensor distances
      // if following the right wall, and away from it, go towards it,
if close to it, go away from it
      if (wall==RIGHT_WALL) {
            if (right_dist<DIP_READ) merge_left();
            else merge_right();
      }
      // if following the left wall, and away from it, go towards it,
if close to it, go away from it
      else {
            if (left_dist<DIP_READ) merge_right();
            else merge_left();
      }
      // if wall ahead, wait 1 second and check again
      if (!FRONT_SENSOR) {
            delay_secs(1);
            // if following the right wall, turn left
            if (!FRONT_SENSOR) {
                  if (wall==RIGHT_WALL) while (!FRONT_SENSOR)
turn_left(1);
                  // if following the left wall, turn right
                  else while (!FRONT_SENSOR) turn_right(1);
            }
      }
}



// search for a wall to follow when started up
void search () {
      dist_read();  // read distances from walls

      // if right wall is closer than 50cm, go along right wall
      if (right_dist<=50) {
            wall = RIGHT_WALL;      // following right wall
      }
      // if left wall is closer than 50cm, go along left wall
      else if (left_dist<=50) {
            wall = LEFT_WALL; // following left wall
      }
      // if no wall is closer than 50cm, go forward
      else {      // go forward until left or right wall is 35cm away,
or reaches a wall ahead
            while (FRONT_SENSOR && right_dist>=35 && left_dist>=35) {
                  go_forward();
                  delay(10000);
                  dist_read();
            }
            // if right wall is closer than 40cm, follow right wall
            if (FRONT_SENSOR && right_dist<=40) {
                  wall=RIGHT_WALL;  // following right wall
            }
            // if left wall is closer than 40cm, follow left wall
            else if (FRONT_SENSOR && left_dist<=40) {
                  wall = LEFT_WALL;  // following left wall
            }
            // if wall ahead, turn right and follow left wall
            else {
                  // go forward for 1 second
                  go_forward();
                  delay_secs(1);
                  // turn right for 1 second,then go forward for 1
second until no wall ahead
                  while (!FRONT_SENSOR) {
                        turn_right(1);
                        go_forward();
                        delay_blinkR_secs(1);
                  }
                  wall=LEFT_WALL;  // follow left wall
            }
      }
}
```

```
// turn vehicle rogjt slightly by turning left motor clockwise and
right motor on and off
// turn right LED on
void merge_right() {
      go_forward();
      R_off();
      R_LED=LED_HIGH;
      L_LED=LED_LOW;
      delay(1500);
      go_forward();

}

// turn vehicle left slightly by turning right motor clockwise and left
motor on and off
// turn left LED on
void merge_left() {
      go_forward();
      L_off();
      R_LED=LED_LOW;
      L_LED=LED_HIGH;
      delay(1500);
      go_forward();
}



// turn vehicle right by turning left motor clockwise, and right
counter-clockwise
void turn_right(int n) {
      R_backward();
      L_forward();
      delay_blinkR_secs(n);
}


// turn vehicle left by turning right motor clockwise, and left
counter-clockwise
void turn_left(int n) {
      R_forward();
      L_backward();
      delay_blinkL_secs(n);
}

// turn both motors clockwise
void go_forward() {
      R_forward();
      L_forward();

}


// turn both motors counter-clockwise
void go_backward() {
      R_backward();
      L_backward();
}
```

```
// read the values from the ADC (which is connected to both sensors)
void dist_read() {
            // read from right sensor
        P3_5=0;                               //init ALE to low
        P3_4=0;                             // select ADC input 0 (right
sensor)
        P3_5=1;                             // pulse ALE L->H->L
        delay(500);                   // delay for 5ms to give ADC time to
complete conversion
        P3_5=0;
        ADC_READS1=P1;              //Read ADC output

            // read from left sensor
        P3_4=1;                             //select ADC input 1 (left sensor)
        P3_5=1;                             // pulse ALE L->H->L
        delay(500);                   // delay for 5ms to give ADC time to
complete conversion
        P3_5=0;
        ADC_READS2=P1;              //Read ADC output

            right_dist=calc_distS1(ADC_READS1);  // look up distance
for right sensor reading
            left_dist=calc_distS2(ADC_READS2);  // look up distance for
left sensor reading
}

// right motor turns clockwise
void R_forward(void) {
      R_ENABLE=1;                   // right enable on
    R_DIRECTION=0;                  // right direction forward
}


// right motor turns counter-clockwise
void R_backward() {
      R_ENABLE=1;             // right enable on
      R_DIRECTION=1;                // right direction backward
}

// turn right motor off by setting right enable low
void R_off() {
      R_ENABLE=0;             // right enable off
}


// left motor turns clockwise
void L_forward() {
      L_ENABLE=1;             // left enable on
      L_DIRECTION=0;                // left direction forward
}
```

```
// left motor turns counter-clockwise
void L_backward() {
      L_ENABLE=1;       // left enable on
      L_DIRECTION=1;          // left direction backward

}

// turn left motor off by setting left enable low
void L_off() {
      L_ENABLE=0;
}


// external interrupt function, initialized and activated but not used
void EXO_int (void) interrupt 0 {


}


// Timer ISR (interrupt service routine) is activated every 100us
void Timer0_ISR (void) interrupt 1 {
      // reset timer
      TH0  = 0XFF;            //reset timer MSB
      TL0  = 0X9C;            //reset timer LSB
      i++;  // global variable i is incremented every 100us
}


// look up table for right sensor, red sensor
unsigned int calc_distS1(unsigned int val) {
      if (val>=0xFF) return 6;
      if (val>=0xFF) return 7;
      if (val>=0xEA) return 8;
      if (val>=0xC8) return 9;
      if (val>=0xB2) return 10;
      if (val>=0xA0) return 11;
      if (val>=0x91) return 12;
      if (val>=0x85) return 13;
      if (val>=0x7B) return 14;
      if (val>=0x72) return 15;
      if (val>=0x6A) return 16;
      if (val>=0x63) return 17;
      if (val>=0x5D) return 18;
      if (val>=0x58) return 19;
      if (val>=0x53) return 20;
      if (val>=0x4F) return 21;
      if (val>=0x4B) return 22;
      if (val>=0x47) return 23;
      if (val>=0x44) return 24;
      if (val>=0x41) return 25;
      if (val>=0x3E) return 26;
      if (val>=0x3C) return 27;
      if (val>=0x3A) return 28;
      if (val>=0x37) return 29;
      if (val>=0x35) return 30;
      if (val>=0x34) return 31;
      if (val>=0x32) return 32;
      if (val>=0x30) return 33;
      if (val>=0x2F) return 34;
      if (val>=0x2D) return 35;
      if (val>=0x2C) return 36;
      if (val>=0x2B) return 37;
      if (val>=0x29) return 38;
      if (val>=0x28) return 39;
      if (val>=0x27) return 40;
      if (val>=0x26) return 41;
      if (val>=0x25) return 42;
      if (val>=0x24) return 43;
      if (val>=0x23) return 44;
      if (val>=0x22) return 45;
      if (val>=0x22) return 46;
      if (val>=0x21) return 47;
      if (val>=0x20) return 48;
      if (val>=0x1F) return 49;
      if (val>=0x1F) return 50;
      if (val>=0x1E) return 51;
      if (val>=0x1D) return 52;
      if (val>=0x1D) return 53;
      if (val>=0x1C) return 54;
      if (val>=0x1C) return 55;
      if (val>=0x1B) return 56;
      if (val>=0x1B) return 57;
      if (val>=0x1A) return 58;
      if (val>=0x1A) return 59;
      if (val>=0x19) return 60;
      if (val>=0x19) return 61;
      return 62;
}



// look up table for left sensor, blue sensor
unsigned int calc_distS2(unsigned int val) {
      if (val>=0xFF) return 6;
      if (val>=0xFF) return 7;
      if (val>=0xDB) return 8;
      if (val>=0xC0) return 9;
      if (val>=0xAB) return 10;
      if (val>=0x9A) return 11;
      if (val>=0x8C) return 12;
      if (val>=0x80) return 13;
      if (val>=0x77) return 14;
      if (val>=0x6E) return 15;
      if (val>=0x67) return 16;
      if (val>=0x60) return 17;
      if (val>=0x5A) return 18;
      if (val>=0x55) return 19;
      if (val>=0x51) return 20;
      if (val>=0x4C) return 21;
      if (val>=0x49) return 22;
      if (val>=0x45) return 23;
      if (val>=0x42) return 24;
      if (val>=0x3F) return 25;
      if (val>=0x3D) return 26;
      if (val>=0x3A) return 27;
      if (val>=0x38) return 28;
      if (val>=0x36) return 29;
      if (val>=0x34) return 30;
      if (val>=0x32) return 31;
      if (val>=0x2E) return 32;
      if (val>=0x2F) return 33;
      if (val>=0x2E) return 34;
      if (val>=0x2C) return 35;
      if (val>=0x2B) return 36;
      if (val>=0x2A) return 37;
      if (val>=0x28) return 38;
      if (val>=0x27) return 39;
      if (val>=0x26) return 40;
      if (val>=0x25) return 41;
      if (val>=0x24) return 42;
      if (val>=0x23) return 43;
      if (val>=0x23) return 44;
      if (val>=0x22) return 45;
      if (val>=0x21) return 46;
      if (val>=0x20) return 47;
      if (val>=0x20) return 48;
      if (val>=0x1F) return 49;
      if (val>=0x1E) return 50;
      if (val>=0x1E) return 51;
      if (val>=0x1D) return 52;
      if (val>=0x1C) return 53;
      if (val>=0x1C) return 54;
      if (val>=0x1B) return 55;
      if (val>=0x1B) return 56;
      if (val>=0x1A) return 57;
      if (val>=0x1A) return 58;
      if (val>=0x19) return 59;
      if (val>=0x19) return 60;
      if (val>=0x18) return 61;
      return 62;
}

// delay function, delays=100usecs*cycles
void delay(int cycles) {
      i=0;
      while (i<cycles) {
            i=i;
      }
    i=0;
}

void delay_secs(int n) {      // delay for n seconds
      int j;
      for (j=0; j< n; j++) delay(10000);
}
```

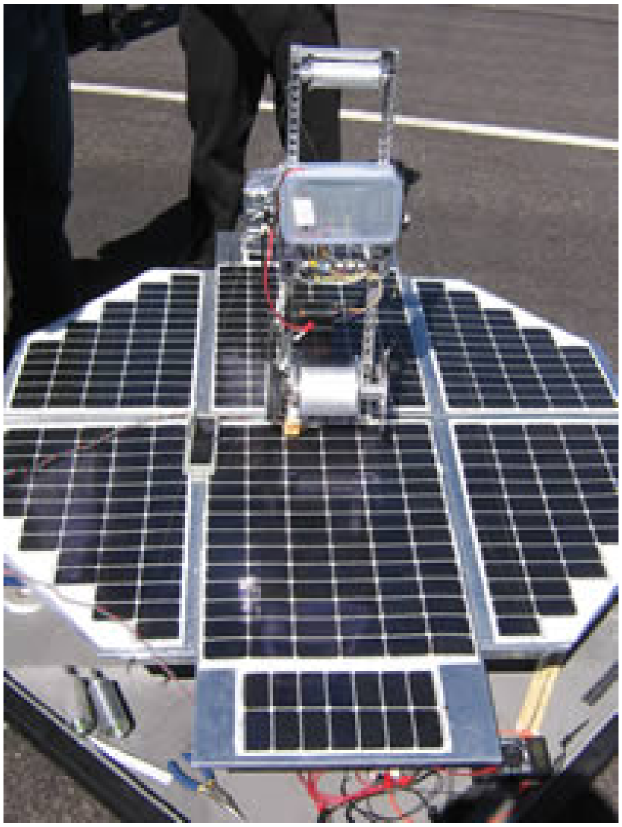
```
// delay for n seconds while blinking right LED
void delay_blinkR_secs(int n) {
      int j;
      L_LED=LED_LOW;  // set left LED low
      for (j=0; j< n; j++) {
            R_LED=!R_LED;  // toggle right LED every quarter second
            delay(2500);
            R_LED=!R_LED;
            delay(2500);
            R_LED=!R_LED;
            delay(2500);
            R_LED=!R_LED;
            delay(2500);
      }
}

// delay for n seconds while blinking left LED
void delay_blinkL_secs(int n) {
      int j;
      R_LED=LED_LOW;  // set right LED low
      for (j=0; j< n; j++) {
            L_LED=!L_LED;  // toggle left LED every quarter second
            delay(25000);
            L_LED=!L_LED;
            delay(2500);
            L_LED=!L_LED;
            delay(2500);
            L_LED=!L_LED;
            delay(2500);
      }
}
```

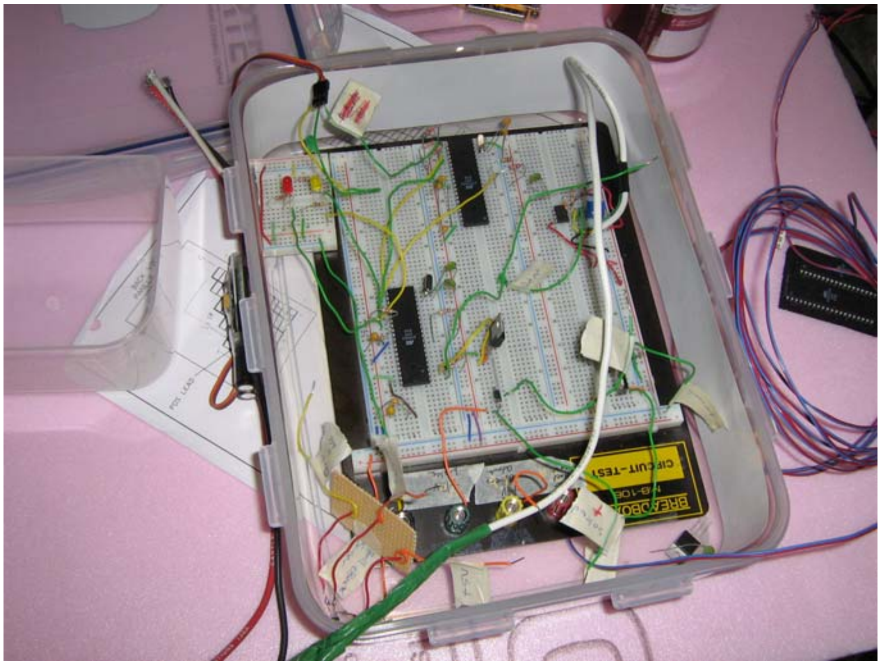
9) Space Elevator

This project was done for the space-elevator competition. The elevator climbs when the beam is on and hasn’t reached the top, breaks when it reaches the top, and descends when the beam is off. Google “elevator2010” to read more about it. And youtube “Univ. of Saskatchewan space elevator qualifying run” for the video of the elevator in New Mexico 2006. We placed first for the second time, which was our second appearance and the second year the club has existed for.

USST- SPACE ELEVATOR 2006



The Circuit Board



```
/*USST
* October 2006
*  To compile: sdcc code.c -llcd
*  To burn: burn8953 code.ihx
*
*  Bashir Oyetunji
*/


#include <8051.h>
#include <stdio.h>

int i;
int speed_c; // counting variable
int speed_temp; // temporary variable to calculate speed;
int speed_angular; // number of triggers per rotation i.e. speed of
elevator in triggers/rotation
int cur_speed; // current speed state
int state; // current elevator state







// defines
#define MIN 0
#define MID 1
#define MAX 2


#define IDLE 0          // led pin out P2_5
#define ASCEND 1  // led pin out P2_6
#define DESCEND 2 // led pin out P2_7

#define HIGH 1
#define LOW 0

#define BEAM P1_0 // HIGH if beam is on
#define SOLENOID P1_1   // outport, normally LOW, solenoid off
#define TOUCH_SENSOR P1_2 // HIGH if not at top



// function declaration
void delay(int);
void delay_secs(int);
void ascend_sp(int);
void status();
void program();


void main(void) {

      // Interrupt Initialization
      IT0 = 1;                // make interrupt edge triggered
      IT1 = 1;
      IE0 = 0;
      EX0 = 1;                // enable external interrupt 0
      EX1 = 1;
      EA  = 1;                // global interrupts enable

// Timer Initialization
      TMOD = 0x01;            // sets timer to a 16 bit continuous
timer
      TH0  = 0XFF;            // Initialize timer0 MSB (timer for
100us)
      TL0  = 0X9C;            // Initialize timer0 LSB
      ET0  = 1;               // Enable Timer0 overflow interrupt
      TR0  = 1;               // Start Timer0

      //Variable Initializations
      i=0;
      speed_c=0;
      speed_temp=0;
      speed_angular=0;

      cur_speed=MIN;
      state=IDLE;
      ascend_sp(MIN);
      delay_secs(7);
      //printf("start\n");

      // set system to idle
      state=IDLE;
      status();

      // set motor control interrupt pin high
      P2_0=1;

      // Loop the following code forever
      for(;;)
            {
                  program();
            }
}

void program() {
      while (BEAM==HIGH) {
            while (TOUCH_SENSOR==HIGH && BEAM==HIGH) { // not at top
                  state=ASCEND;
                  status();
                  if (cur_speed != MAX)   ascend_sp(MAX);
            }
            while (TOUCH_SENSOR==LOW && BEAM==HIGH) {
                  state=IDLE;
                  status();
                  // wait and do nothing till beam goes low or doesnt
touch
            }
            state=ASCEND;
            status();
      }
      if (BEAM==LOW && state==ASCEND) { // go to idle, wait 5 secs
            state=IDLE;
            status();
            ascend_sp(MIN); //brake with motors
            SOLENOID=LOW;     //Brake with solenoid
            delay_secs(5); // pause for 5 secs
            state=DESCEND;
            status();
      }
      while (BEAM==LOW && state==DESCEND) {  //descend with speed
control
            if (speed_angular<=10) { // speed control
                  SOLENOID=LOW;     // turn solenoid on for 1 sec
                  delay_secs(1);
                  SOLENOID=HIGH;
            }
      }
}


// set the status LED and speed of elevator
void status() {
      if (state==IDLE) {
            //printf("IDLE\n");
            P2_5=1;
            P2_6=0;
            P2_7=0;
            // set ascend speed to MIN or brake
            ascend_sp(MIN);
            //turn solenoid on
            SOLENOID=LOW;
      }
      else if (state==ASCEND) {
            //printf("ASCEND\n");
            P2_5=0;
            P2_6=1;
            P2_7=0;
            // set ascend speed to MAX
            ascend_sp(MAX);

      }
      else if (state==DESCEND) {
            //printf("DESCEND\n");
            P2_5=0;
            P2_6=0;
            P2_7=1;
            // set ascend speed to MIN or brake
            ascend_sp(MIN);
            //turn solenoid off
            SOLENOID=HIGH;
      }
}
```

```
// Timer ISR (interrupt service routine) is activated every 100us
void Timer0_ISR (void) interrupt 1 {
      // reset timer
      TH0  = 0XFF;            //reset timer MSB
      TL0  = 0X9C;            //reset timer LSB
      i++;  // global variable i is incremented every 100us
      speed_c++;
      if (speed_c>=1000) {
            speed_temp++;
            speed_c=0;
      }
}

// External interrupt 1 to calculate speed of elevator
void EXO_int (void) interrupt 0 {
      speed_angular=speed_temp;
      //printf("%d tr/r\n", speed_angular);
      speed_c=0;
      speed_temp=0;
      //speed is between >2 and <5 triggers*
}


// External interrupt 2, does nothing
void EXO_int1 (void) interrupt 2
{
      //printf("ext1\n");
}



void ascend_sp(int sp) {
      if (sp==MIN) {
      //    printf("min\n");
            // set pins
            P2_2=0;
            P2_3=0;
            // latch data
            P2_0=0;
            delay(5000);
            P2_0=1;
            cur_speed=MIN;
            // always stay 6 secs at min speed

      }
      else if (sp==MID) {
      //    printf("mid\n");
            // set pins
            P2_2=1;
            P2_3=0;
            // latch data
            P2_0=0;
            delay(5000);
            P2_0=1;
            cur_speed=MID;

      }
      else if (sp==MAX) {
      //    printf("max\n");
            // set pins
            P2_2=0;
            P2_3=1;
            // latch data
            P2_0=0;
            delay(5000);
            P2_0=1;
            cur_speed=MAX;
      }
}


// delay function, delay=100usecs*cycles
void delay(int cycles) {
      i=0;
      while (i<cycles) {
            i=i;
      }
    i=0;

}

// delay for n seconds
void delay_secs(int n) {
      int j;
      for (j=0; j< n; j++) delay(10000);
}
```

I hope you found this guide helpful. If you have any questions, you can contact me at bashir.oye@ieee.org .